الجمهورية الجزائرية الديمقراطية الشعبية
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
وزارة التكوين والتعليم المهنيين
Ministère de la Formation et de l'Enseignement Professionnels

Centre National de l'Enseignement Professionnel à Distance (CNEPD)

المركز الوطني للتعليم المهني عن بعد
(م.و.ت.م.ب)

COURS D'ALGORITHME

SÉRIE 05

LES SOUS PROGRAMMES

OBJECTIF PÉDAGOGIQUE : À la fin de cette série, le stagiaire doit être capable de créer un sous programmes algorithmique.

PLAN DE LA LEÇON :

INTRODUCTION :

La plupart des problèmes auxquels est confronté le programmeur peuvent se décomposer en sous problèmes moins complexes, dont la résolution est pratiquement aisée. L'algorithme de résolution est alors composé d'une suite ordonnée d'algorithmes de résolution de chaque sous problème. Ces derniers sont appelés sous programmes ou actions paramétrées et qui feront l'objet de notre série.

Exemple :

Soient L1, L2 et L3 trois listes contenant les notes obtenues par une promotion, respectivement à la première, deuxième et troisième épreuve. Ces listes sont données dans un ordre quelconque. On veut calculer la moyenne de tous les étudiants de cette promotion. Paramétrée

Cette opération peut être effectuée par le tri des trois listes (par ordre alphabétique sur le champ NOM), ensuite le calcul de la moyenne des trois notes pour chaque élément de la liste. L'algorithme général comportera donc les traitements suivants :

1. TRIER L1 <* Sous programme1*>
2. TRIER L2 <* Sous programme2*>
3. TRIER L3 <* Sous programme3*>
4. CALCULER LA MOYENNE DES NOTES <*Sous programme4*>

Si l'on observe de trop prés les trois premières parties de l'algorithme, on se rend compte rapidement qu'il s'agit de la même opération de base à la différence que dans chaque liste les noms n'apparaissent pas dans le même ordre et que les notes pour un même étudiant sont différentes car il s'agit de trois épreuves différentes.

Pour résoudre cette répétition , la solution informatique consiste à déclarer dans l'algorithme, l'ensemble des actions et des objets communs d'une manière adéquate pour pouvoir leur faire appel ailleurs dans l'algorithme autant de fois que l'on désire c'est ce qu'on appelle les actions paramétrées ou sous programmes.

La structure de l'algorithme précédent devient alors :

{Déclaration des variables de l'algorithme} ;
{Déclaration de l'action paramétrée qui fait le tri} ;
{Bloc1 d'instructions qui lisent les trois listes} ;
{Appel de l'action paramétrée pour trier L1} ;
{Appel de l'action paramétrée pour trier L2} ;
{Appel de l'action paramétrée pour trier L3} ;
{Bloc2 d'instructions qui calculent la moyenne des trois notes} ;

Remarque : L'appel de l'action paramétrée est une opération simple que nous verrons plus loin.

I- DÉFINITION DE L'ACTION PARAMÉTRÉE :

L'action paramétrée est une suite d'actions regroupées en une seule action, à laquelle est associé un identificateur qui permet de la référencer en tout point de l'algorithme.

L'action paramétrée est donc, un outil très puissant pour la conception d'algorithmes et ce pour une meilleur lisibilité d'une part et une structuration facilitant la mise au point des algorithmes d'autre part.

II- CONSTITUANTS DE L'ACTION PARAMÉTRÉE :

Une action paramétrée est constituée :

- **D'un en-tête** : qui correspond au nom de l'action suivi d'une suite de paramètres qui spécifient les relations avec l'extérieur.

- **D'un corps :** qui précise le contenu de l'action paramétrée. Il est constitué d'une partie déclarations d'objets propres à l'action paramétrée, et d'une partie actions.

Elle se schématise par :

Action < Identificateur > (< Spécification des paramètres >) ;
Début
{Partie Déclarations} ;
{Partie Actions} ;
Fin ;

Remarque :

- A la différence de l'algorithme, l'action paramétrée se termine par Fin ; au lieu de Fin.
- Une action paramétrée peut être appelée par un algorithme (dit appelant) ou une autre action paramétrée.

L'utilisation des actions paramétrées en algorithmique, consiste donc à déclarer toutes les actions paramétrées dans la partie déclarations de l'algorithme et à faire leurs appels dans la partie actions.

Exemple :

Soit ALGO1 un algorithme constitué de deux actions paramétrées Ap1 et Ap2, il sera représenté par :

Algorithme ALGO1 ;
Début
{Déclarations des variables de l'algorithme} ;
{Déclaration de Ap1 } ;
{Déclaration de Ap2 } ;

{Bloc1d'instructions} ;
{Appel deAp1} ;
{Bloc2d'instructions} ;
{Appel de Ap2} ;
{Bloc3d'instructions} ;
Fin.

III- OBJETS MANIPULÉS PAR L'ACTION PARAMÉTRÉE :

Les objets ou les variables manipulés par l'action paramétrée sont :

- Soit des paramètres spécifiés dans l'en-tête de l'action paramétrée ;
- Soit des objets déclarés dans la partie déclarations de l'action paramétrée et qui s'appellent objets locaux ;
- Soit des objets déclarés dans l'algorithme appelant et qui sont différents des paramètres de l'en-tête de l'action paramétrée et qui s'appellent objets globaux.

Exemple :

Soient V1 et V2 deux variables entières utilisées par l'action paramétrée 'Action1'. Les trois cas possibles sont :

1er cas : V1 et V2 sont des paramètres de 'Action1'

```
ALGORITHME  Obj_act ;

Début
{Déclarations des variables de l'algorithme Obj_act } ;

Action  Action1 (V1, V2 : Entier) ;    (* déclaration de l'action paramétrée *)
      Début
      {Déclarations des variables locales de Actionl } ;
      {Partie actions de Action 1} ;
      Fin ;
{Partie actions de l'algorithme Obj_act} ;
Fin.
```

2ème cas : V1 et V2 sont des variables locales de 'Action1'

```
ALGORITHME  Obj_act ;

Début
{Déclarations des variables de l'algorithme Obj_act } ;
```

Action Action1 (Liste des paramètres différents de V1etV2) ;

Début
V1, V2 : Entier ;
{Déclarations des autres variables locales d'Action1} ;
{Partie actions de Action1} ;
Fin ;

{Partie actions de l'algorithme Obj_act } ;
Fin.

3ème cas : V1 et V2 sont des variables globales de 'Obj_act'

ALGORITHME Obj_act ;

Début
V1, V2 : Entier ;
{Déclarations des autres variables de l'algorithme Obj_act} ;

Action Action1 (Liste des paramètres différents de V1et V2) ;

Début
{Déclarations des variables de Actionl } ;
{Partie actions de Actionl } ;
Fin ;

{Partie actions de l'algorithme Obj_act } ;
Fin.

Remarque :

Les déclarations des objets locaux et des objets globaux se font respectivement à la partie déclarations de l'action paramétrée et à la partie déclarations de l'algorithme appelant. La différence qui réside entre ces deux catégories d'objets, est que les objets globaux sont utilisés partout dans l'algorithme appelant et dans les actions paramétrées qu'il contient ;

Par contre les objets locaux ne sont reconnus que dans le corps de l'action elle même, et toute référence à eux en dehors de l'action paramétrée est inexacte. Dans ce qui suit, nous nous intéressons beaucoup plus aux objets paramètres.

III- LES OBJETS PARAMÉTRÉS :

Les paramètres sont des variables introduites dans l'en-tête de l'action paramétrée, assurant ainsi la transmission d'informations entre l'algorithme appelant et l'action appelée.

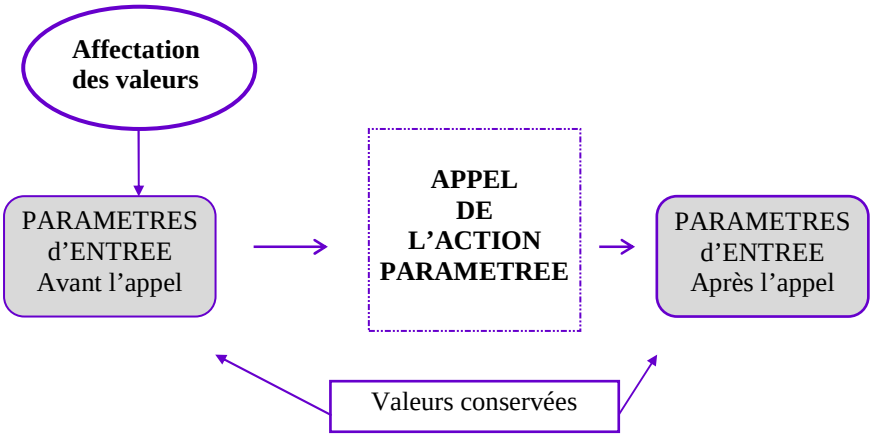
Les paramètres introduits dans l'en-tête de l'action paramétrée, lors de sa déclaration sont appelés paramètres formels. Lors de l'appel de l'action paramétrée, les valeurs substituées aux paramètres formels sont appelés paramètres effectifs.

Nous distinguons trois types de paramètres : les paramètres d'Entrée (ou données), les paramètres de Sortie (ou résultats) et les paramètres d'Entrée/Sortie (ou données/résultats).

1- Paramètres d'entrée :

Ils constituent la (ou les) condition (s) d'exécution de l'action paramétrée. Ce sont des objets que l'action paramétrée utilise mais qu'elle ne modifie pas pour l'algorithme appelant :

C'est à dire que les valeurs qu'on leur affecte avant l'appel, restent inchangées lors du retour à l'algorithme appelant.



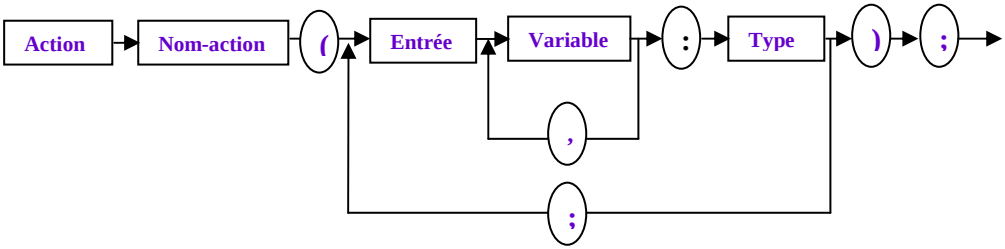
Syntaxe : La déclaration des paramètres d'entrée se fait par **:**

Action <nom de l'action>(Entrée<liste$_1$ des var> : type$_1$; ... Entrée <liste$_n$ des var> **:**type$_n$) **;**

Où : Action est le mot réservé qui déclare toute action paramétrée ;
Nom de l'action est son identification ;

Entrée est le mot réservé qui indique que le mode de transmission des paramètres est : Entrée ;
liste$_i$ des var est une liste de variables séparées par des virgules ;
type$_i$ est un type associé à une liste de variables.

Et qui est représentée par le diagramme :



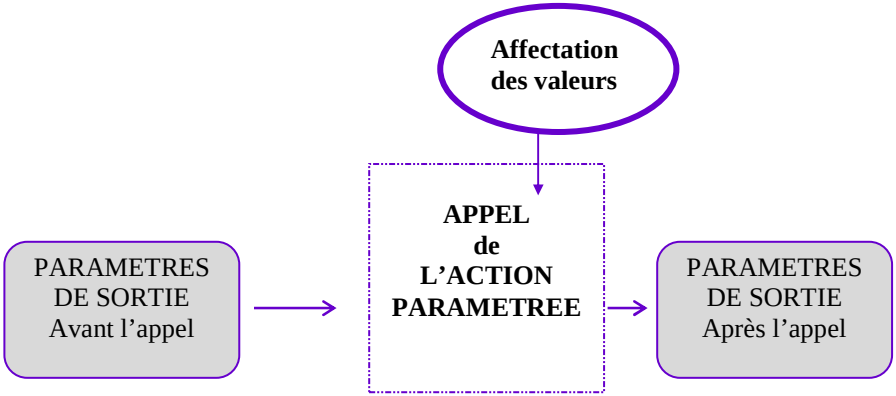
Exemple :

Actions_mot (Entrée mot1 : Chaîne (3) ; Entrée mot2 : Chaîne (10)) ;

Action produit (Entrée a1 : Tableau (10,20) ; Entrée a2 : Tableau (20,30)) ;

2- Les paramètres de sortie :

A l'inverse des paramètres d'entrée, ces paramètres ne sont destinataires d'aucune valeur avant l'appel de l'action paramétrée. C'est au cours de l'exécution de l'action paramétrée que des valeurs leur sont affectées. Ce sont des objets globaux puisque leurs valeurs sont atteintes par l'algorithme appelant qui pourra les modifier.

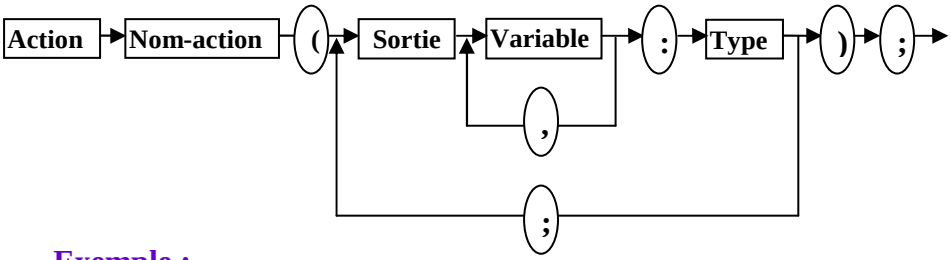


Syntaxe : La déclaration des paramètres de sortie se fait par :

Action <nom de l'action>(Sortie<liste$_1$ des var> : type$_1$; ... Sortie <liste$_n$des var> : type$_n$) ;

Où : Sortie est le mot réservé qui indique que le mode de transmission des paramètres est : Sortie ;

Et qui est représentée par le diagramme **:**

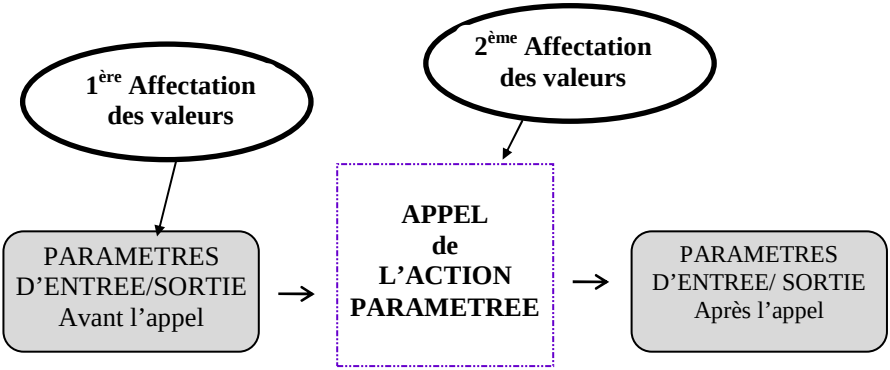


Exemple :

Action Racine (Entrée x : Réel ; Sortie y: Réel) ;

3- Les paramètres d'entrée/sortie :

Ce sont des paramètres que l'action paramétrée utilise et modifie. Ils ont initialement (avant l'exécution de l'action paramétrée) des valeurs que l'action paramétrée peut leur définir de nouvelles valeurs lors de son activation. Ce sont donc des objets globaux.

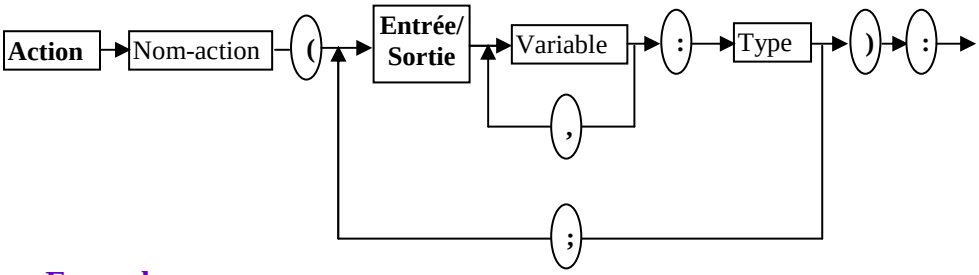


Syntaxe : La déclaration des paramètres d'entée/sortie se fait par :

Action <nom de l'action> (Entrée/Sortie <liste$_1$ des var> : type$_1$; ...
Entrée/Sortie <liste$_n$ des var> : type$_n$) ;

Où : Entrée/Sortie est le mot réservé qui indique que le mode de transmission des paramètres est : Entrée/Sortie ;

Et qui est représentée par le diagramme :



Exemple :

Action Tri (Entrée ordre : Entier ; Entrée/Sortie v: Tableau (100) Entier) ;

Cette action réalise le tri ou le classement des éléments d'un tableau de type entier, dans l'ordre croissant ou décroissant suivant la valeur du paramètre 'ordre'. Donc les éléments de v au départ (en entrée) sont quelconques et après l'appel de cette action (en sortie) ils sont ordonnés.

V- ÉCRITURE D'ACTIONS PARAMÉTRÉES :

Comme nous l'avons vu, une action paramétrée est constituée d'un en-tête, d'une partie déclarations et d'un bloc d'instructions qui représentent un ensemble d'opérations bien spécifique. L'écriture donc, d'une action paramétrée nécessite tout d'abord de définir la (ou les) opération (s) qu'elles réalisent, ensuite de décrire l'en-tête par la donnée du nom de l'action et de l'ensemble de ses paramètres ainsi que leur type et mode de transmission et enfin, de déclarer tout ses objets locaux et ses instructions.

On a donc défini, trois modes de transmission des paramètres entre l'algorithme appelant et l'action paramétrée appelée :

- Mode 'Entrée'
- Mode 'Sortie'
- Mode 'Entrée/Sortie'

Exemple :

Soient V1 et V2 deux vecteurs ayant respectivement M et N composantes. On veut réaliser la fusion de ces vecteurs en un vecteur V3, dont les éléments soient rangés par ordre croissant.

Cette opération peut être effectuée par le tri des deux vecteurs, ensuite leur fusion en respectant le tri. L'algorithme général comportera donc les traitements suivants :

1- Trier V1
2- Trier V2
3- Trier V3

et sera donc **:**

```
Algorithme TRI-FUS ;
Début
V1, V2 : Tableau (100) Entier ;
V3 : Tableau (200) Entier ;
I, J, K, M, N, Z, NB1, NB2: Entier;

Lire (N, M) ;  (* Lecture des tailles de V1  et  V2 *)
```

```
Pour I := 1   à   M
      Faire  Lire (V1(I)) ;
      Fait ;
Pour I := 1   à   N
      Faire  Lire (V2(I)) ;
  Fait ;

(* TRI DU VECTEUR V1 *)
NB1 := 0 ;
Pour I := 1  à  M-1
      Faire   Pour  J := I + 1  à  M
                  Faire     Si  V1(I) > V1(J)    (* Permutation de
V1(I) et V1(J) *)
AlorsZ: =V1 (I);
V1 (I):= V1 (J);
V1(J) := Z ;
          NB1 :=NB1+1 ;(*Compte les permutations *)
                        Fsi ;
                  Fait ;
      Fait ;
Pour  I := 1  à  M
       Faire    Ecrire (V1(I)) ;
       Fait ;
(* TRI DU VECTEUR V2 *)
NB2 := 0 ;
Pour I := 1  à  N-1
      Faire   Pour  J := I + 1  à  N
                  Faire     Si  V2(I) > V2(J)    (* Permutation de
V2(I) et V2(J) *)
AlorsZ: =V2 (I);
             V2 (I):= V2 (J);
V2(J) := Z ;
NB2 :=NB2+1 ;(*Compte les permutations *)
                        Fsi ;
                  Fait ;
      Fait ;

     Pour  I := 1  à  N
       Faire    Ecrire (V2(I)) ;
```

```
        Fait ;

(* Fusion de V1 et V2 *)
I := 1 ; J := 1 ; K := 1 ;
Tant que  I <= M   Et   J <= N
            Faire   Si  V1(I) < V2(J)
AlorsV3(K) := V1(I) ;
I: = I+1
Sinon V3(K) := V2(J) ;
                                    J := J+1 ;
                    Fsi ;
              K := K+1 ;
              Fait ;
Tant que  I <= M  (*Copie les éléments de V1 qui sont *)
  Faire   V3(K) := V1(I) ; (*supérieurs à ceux de V2 *)
                    I := I + 1 ;
                    K := K + 1 ;
            Fait ;
Tant que  I <= N (*Copie les éléments de V2 qui sont *)
 Faire   V3(K) := V2(I) ; (*supérieurs à ceux de V1 *)
                    I := I + 1 ;
                    K := K + 1 ;
            Fait ;
L := M + N ;        (* Calcul de la taille de V3 *)
Pour  I := 1  à  L
        Faire    Ecrire (V3(I)) ;
        Fait ;

Ecrire ('Il y a eu ',NB1, 'Permutations dans V1') ;
Ecrire ('Il y a eu ',NB2, 'Permutations dans V2') ;

Fin.
```

Une telle écriture de l'algorithme rend difficile sa lisibilité, malgré l'aide des commentaires, et on constate qu'un même problème a été résolu deux fois : Le tri d'un vecteur. Les deux blocs de tri de V1 et V2 sont identiques à l'exception des noms des variables. Pour pallier à ces inconvénients, on introduit deux actions paramétrées : Trier et fusion :

- L'action Trier permet de trier un vecteur A donné de taille D, les résultats sont le vecteur A réordonné et le nombre de permutations P. On l'écrira donc en fonction des paramètres :

  1- A : paramètre d'Entrée/Sortie
  2- D : paramètre d'Entrée
  3- P : paramètre de Sortie

- L'action Fusion fusionne les deux vecteurs A et B respectivement de tailles D1 et D2, en un vecteur C de taille D3 = D1 + D2. Ses paramètres sont :

  1. A, B, D1, D2 : paramètres d'Entrée
  2. C, D3 : paramètres de sortie

L'algorithme sera donc :

```
Algorithme Tri-Fus ;
Début
V1, V2 : Tableau (100) Entier ;
V3 : Tableau (200) Entier ;
M, N, L, NB1, NB2, I:Entier;
Action Trier (Entrée D : Entier ; Entrée/Sortie A : Tableau
(100) Entier ; Sortie P : Entier);

Début
I1, I2, Z:Entier;
P := 0 ;
Pour  I1 := 1  à  D -1
      Faire   Pour  I2 := I1 + 1  à  D
Faire    Si  A(I1) > A(I2)   (* Permutation de A(I1) et A(I2) *)
 Alors    Z := A(I1) ;
        A(I1) := A(I2) ;
        A(I2) := Z ;
        P :=P+1 ;    (*Compte les permutations *)
                              Fsi ;
                      Fait ;
         Fait ;
```

```
Pour  I1 := 1  à  D
      Faire    Ecrire (A(I1)) ;
      Fait ;
Fin ;

Action  Fusion (Entrée A, B : Tableau (100) Entier ; Entrée D1,
D2 : Entier ;
 Sortie C : Tableau (200) Entier ; Sortie D3 : Entier ;) ;
Début
I, J, K : Entier ;
I :=1 ; J :=1 ; K := 1 ;
Tant que  I <= D1    Et    J <= D2
            Faire  Si  A(I) < B(J)
AlorsC (K):= A (I);
I: = I + 1

Sinon  C(K) := B(J) ;
                                  J := J + 1 ;
                  Fsi ;
             K := K + 1 ;
             Fait ;
Tant que I <= D1
Faire C (K):= A (I);
I: = I + 1;
K := K + 1 ;
           Fait ;
Tant que J <= D2
           Faire  C(K) := B(J) ;
                  J := J + 1 ;
                  K := K + 1 ;
           Fait ;
D3 := D2 + D1 ;
Pour  I := 1  à  D3
      Faire   Ecrire ( C(I) ) ;
      Fait ;
Fin ;

(* Programme Principal *)
Lire (M, N) ;
Pour  I := 1  à  M
```

```
        Faire     Lire (V1(I)) ;
        Fait ;

      Pour  I := 1  à  N
        Faire     Lire (V2(I)) ;
        Fait ;
Trier (M, V1, NB1) ;
Trier (M, V2, NB2) ;
Fusion (V1, V2, M, N, V3, L) ;
Fin.
```

Remarque:

1. Dans l'action paramétrée Trier, I1 et I2 ne sont pas des paramètres, se sont des variables locales et leur domaine de validité se limite au corps de cette action paramétrée.

2. Dans l'action paramétrée Fusion, la variable I est locale et n'a rien à voir avec la variable globale I déclarée au début de l'algorithme appelant. La valeur qui lui sera affectée pendant l'activation de cette action, n'est pas conservée au retour à l'algorithme (avant l'appel, la variable globale I contient la valeur N et après l'appel elle ne subi aucune modification ; Par contre la variable locale à la procédure Fusion se voit affecter la valeur N + M).

3. Dans la déclaration des paramètres de l'action paramétrée, on convient de donner suivant l'ordre suivant :

   En premier lieu, les paramètres d'entrée ;
   En second lieu, les paramètres d'entrée/sortie ;
   Et en dernier lieu, les paramètres de sortie.

4. En pratique, on distingue deux types d'actions paramétrées :

   - Les actions appelées Procédures et qui possèdent plusieurs sorties.

- Les actions appelées Fonctions et qui possèdent une sortie unique et aucun paramètre d'Entrée/Sortie. Le nom de la fonction joue le rôle du paramètre de sortie.

Il en résulte que la déclaration d'une fonction se fait par :

Fonction <nom-fonction>(<Liste des paramètres d'entrée>) : <Type> ;

Où Type est le type de la fonction

Puisque, le nom de l'action paramétrée joue le rôle du paramètre de sortie, donc il y aura au moins une affectation dans le nom de la fonction au niveau de la fonction elle même.

EXEMPLE :

Soit l'action paramétrée Max qui donne le plus grand nombres de deux réels donnés. Cette action peut être vue comme une fonction ou comme une procédure que nous déclarons :

```
Fonction  Max (Entrée x, y : Réel) : Réel  ;

Début
Si x > y
   Alors      Max := x        (* 1ère affectation dans le nom de la fonction Max *)
   Sinon     Max := y ;       (* 2ème affectation dans le nom de la fonction Max *)
Fsi ;
Fin ;


Procédure  Max (Entrée x , y : Réel ; Sortie M :Réel) ;

Début
Si x > y
   Alors     M := x
   Sinon    M := y ;
Fsi ;
Fin ;
```

Ici la fonction Max est de type réel et elle possède deux paramètres d'entrée (x et y) et retourne le maximum de x et y.

Par contre la procédure Max possède deux paramètres d'entrée (x et y) et un paramètre de sortie (M) qui contiendra le maximum de x et y.

L'appel de cette fonction au niveau d'un algorithme, se fait par simple affectation de cette dernière à une variable de type réel ou dans une expression arithmétique ou logique, bien sûr après substitution des paramètres formels x et y par des valeurs effectives. Ce qui est d'ailleurs traduit par la partie de l'algorithme suivante :

```
A, B, RA : Réel ;    ou encore Si Max (A, B) > 10
Lire (A, B) ; Alors        A := A + 2 ;
RA: = Max (A, B);
Fsi ;
```

RÉSUMÉ :

Après la structuration des données à partir de données simples en structure de tableau et chaînes de caractères, cette série a fait l'objet de structuration d'actions ou composition d'actions à partir d'une ou plusieurs actions de base. C'est le concept fondamental de construction d'algorithmes à partir des actions paramétrées.

Définir une action paramétrée consiste à définir :

- L'ensemble de ses paramètres d'entrée (ou données non modifiables), d'entrée/sortie (ou données/résultats qui changent de valeurs) et de sortie (ou résultats calculés par l'action).

- L'ensemble de ses variables locales qui seront utilisées au cours du traitement à titre temporaire.

- Le bloc de ses instructions qui réalisent un traitement bien spécifique.

L'appel d'une action paramétrée se fait par substitution de ses paramètres formels (décrits lors de sa déclaration) par des paramètres effectifs (variables globales de l'algorithme appelant qui contiennent ou qui reçoivent des valeurs).

L'utilisation d'actions paramétrées, assure une meilleure lisibilité, évite la répétition dans les algorithmes et assure un raisonnement modulaire.

EXERCICES D'APPLICATION:

EXERCICE N°01 :

Écrire la fonction qui détermine si un nombre entier est pair.

EXERCICE N°02 :

On définit une suite d'entiers naturels comme suit :

- Le premier terme U1 sera lu (U1 > 1)
- Si Un est pair Un+1 = Un / 2
- Si Un est impair Un+1 = 3 * Un + 1

Ecrire l'algorithme qui calcule les termes de cette suite. Le calcule s'arrête au rang i tel que Ui = 1.

Cet algorithme devra utiliser la fonction Pair(x) définie dans l'exercice 1.

EXERCICE N°03 :

Soit Vect un tableau de caractères alphabétiques, écrire l'algorithme qui compte le nombre de voyelles qui apparaissent dans ce tableau.

EXERCICE N°04 :

Ecrire une action paramétrée Sous mot qui vérifie qu'un mot donné M0 est un sous-mot d'un mot M1, c'est à dire : M1 = w1 M0 w2 où w1 et w2 sont des mots qui peuvent être vides (ne contiennent aucune lettre).

CORRIGÉ DES EXERCICES D'APPLICATION :

EXERCICE N°01 :

```
Fonction  Pair (Entrée N : Entier) : Booléen ;

Début
X, Y : Entier ;
X := N ;
Y := X DIV 2 ;  (* l'opérateur DIV réalise la division entière *)
X := X - 2 * Y ;
Si   X = 0
     Alors    Pair := Vrai
     Sinon    Pair := Faux ;
Fsi ;
Fin ;
```

EXERCICE N°02 :

```
Algorithme Exo2 ;
Début
I, U : Entier ;
PU : Booléen ;

Fonction  Pair (Entrée N : Entier) : Booléen ;
Début
X, Y : Entier ;
X := N ;
Y := X DIV 2 ; (* l'opérateur DIV réalise la division entière *)
X := X - 2 * Y ;
Si   X = 0
     Alors    Pair := Vrai
     Sinon    Pair := Faux ;
Fsi ;
Fin ;

Lire (U) ;
Ecrire (U) ;
Tant que  U <> 1
          Faire    Si  Pair (U)
```

```
                    Alors    U := U / 2
                    Sinon   U := 3*U + 1 ;
                Fsi ;
                Ecrire (U) ;
        Fait ;
Fin.
```

EXERCICE N°03 :

```
Début
Vect : Tableau (100) Car ;
I, N, NV:Entier;
V:Booléen;

Fonction Voyelle (Entrée C : Car) : Booléen ;

Début
Voyelle := Faux ;
Si  C = 'a'  ou  C = 'e'  ou  C = 'i'  ou  C = 'o'  ou  C = 'u'  ou
C = 'y'
    Alors  Voyelle := Vrai ;
Fsi ;
Fin ;

Lire (N) ;
NV := 0 ;
Pour  I := 1   à  N
    Faire   Lire (Vect (I)) ;
            Si  Voyelle (Vect (I))
                Alors    NV := NV + 1 ;
            Fsi ;
    Fait ;
Ecrire ('Ilya ', NV, 'voyelles' ) ;

Fin ;
```

EXERCICE N°04 :

Fonction Sousmot (Entrée M0 : Tableau (100) Car ;
Entrée M1:Tableau(300) Car ; Entrée D1, D2 : Entier) : Booléen ;
Début
I, J : Entier ;
Sousmot := Faux ;
I := 1 ;
J := 1 ;
Tant que I <= D1 Et J <= D2
Faire Si M0(I) = M1(J)
Alors I := I + 1 ; (* S'il y a égalité on avance*)
J := J + 1 (* dans les deux tableaux *)
Sinon J := J - I + 2 ;(* Sinon on revient au début de *)
I := 1 ; (*M0 et à la position qui suit le début*)
Fsi ; (* de la recherche précédente dans M1 *)
Fait ;
Si I > D1
Alors Sousmot = Vrai ;
Fsi ;
Fin ;

Exécution
Soient M0 = (M, O, I) et M1= (T, M, M, O, M, O, I, S)
D1= 3 et D2=8

I: =1;J: =1 M0 (1) <>M1 (1)
I: =1;J: =2 M0 (1) = M1 (2)
I: =2;J: =3 M0 (2) <>M1 (3)
I: =1;J: =3 M0 (1) = M1 (3)
I: =2;J: =4 M0 (2) = M1 (4)
I: =3;J: =5 M0 (3) <>M1 (5)
I: =1;J: =4 M0 (1) <>M1 (4)
I: =1;J: =5 M0 (1) = M1 (5)
I := 2 ; J := 6 M0(2) = M1(6)
I := 3 ; J := 7 M0(3) = M1(7)
I := 4 ; J := 8 l'expression : I <= D1 est fausse car I=4 et D3=3, on sort donc de la boucle Tant que et Sousmot reçoit la valeur 'Vrai'.